Date=05/08/2020
Lecture By=Shubham Joshi
Subject ⇒ Linked list Problem Solving

IN PREVIOUS LECTURE **(QUICK RECAP) Date-04/08/2020**	In Today's Lecture (Overview)
What is linked list in python Important things What Is Node In Python MCQs	Problem Solving Session Question=1 Find the middle Element of linked List Question=2 Given A linked List You have to reverse it MCQs Questions for self practice

Problem solving
In Today's lecture We Discussed The Problems/questions regarding the linked List

Question=1 Find the middle Element of linked List

Leetcode

```
class ListNode(object):
    def __init__(self, x):
        self.val = x
        self.next = None
class Solution(object):
    def middleNode(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        slow, fast = head, head
        while fast and fast.next:
```

```
            slow, fast = slow.next, fast.next.next
        return slow
```

Geeksforgeeks

```
class Node:
    def __init__(self, value):
        self.data = value
        self.next = None

class LinkedList:

    def __init__(self):
        self.head = None

    # create Node and and make linked list
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    def printMiddle(self):
        temp = self.head
        count = 0

        while self.head:

            # only update when count is odd
            if (count & 1):
                temp = temp.next
            self.head = self.head.next

            # increment count in each iteration
            count += 1

        print(temp.data)
```

```
# Driver code
llist = LinkedList()
llist.push(1)
llist.push(20)
llist.push(100)
llist.push(15)
llist.push(35)
llist.printMiddle()
```

Output

```
E:\Study\Codes\if>C
100
```

Question=2 Given A linked List You have to reverse it

(Complexity Of these codes will be O(n))

Iterative Method

```
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # Function to reverse the linked list
```

```
    def reverse(self):
        prev = None
        current = self.head
        while(current is not None):
            next = current.next
            current.next = prev
            prev = current
            current = next
        self.head = prev

    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Utility function to print the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print(temp.data,)
            temp = temp.next


# Driver program to test above functions
llist = LinkedList()
llist.push(20)
llist.push(4)
llist.push(15)
llist.push(85)

print("Given Linked List")
llist.printList()
llist.reverse()
print ("\nReversed Linked List")
llist.printList()
```

Output

```
Given Linked List
85 15 4 20
Reversed Linked List
20 4 15 85
```

A Simpler and Tail Recursive Method

```
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None


    def reverseUtil(self, curr, prev):

        # If last node mark it head
        if curr.next is None :
            self.head = curr

            # Update next to prev node
            curr.next = prev
            return

        # Save curr.next node for recursive call
        next = curr.next
```

```
        # And update next
        curr.next = prev

        self.reverseUtil(next, curr)


    # This function mainly calls reverseUtil()
    # with previous as None
    def reverse(self):
        if self.head is None:
            return
        self.reverseUtil(self.head, None)


    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Utility function to print the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print(temp.data,)
            temp = temp.next


# Driver program
llist = LinkedList()
llist.push(8)
llist.push(7)
llist.push(6)
llist.push(5)
llist.push(4)
llist.push(3)
```

```
llist.push(2)
llist.push(1)

print ("Given linked list")
llist.printList()

llist.reverse()

print ("\nReverse linked list")
llist.printList()
```

Output

```
Reverse linked list
8
7
6
5
4
3
2
1
```

“Only two Problems were Discussed In this lecture”

MCQs

1.**What is the time complexity for reversing a linked list ?**

A.O(1)

B.O(N)

C.O(logN)

D.O(N2)

2.**Are elements of linked list in continuous memory blocks like arrays ?**

A.no

B.Yes

3.**What is the time complexity to add an element at the start of an array ?**2

A.O(1)

B.O(n)

C.O(n2)

4.**what is the time complexity to add an element at the start of linked list ?**2

A.O(n)

B.O(N2)

C.O(1)

Questions for self practice

1.https://practice.geeksforgeeks.org/problems/nth-node-from-end-of-linked-list/1

2.https://practice.geeksforgeeks.org/problems/count-nodes-of-linked-list/1

3.https://practice.geeksforgeeks.org/problems/reverse-a-linked-list/1